design ideas

Edited by Bill Travis and Anne Watson Swager

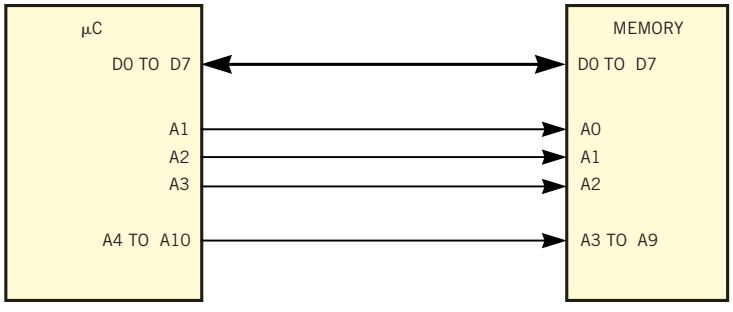
Access odd memory locations without hardware

Sorin Zarnescu, NEC Electronics, Santa Clara, CA

Some RISC controllers, like the NEC V850 family, use an internal 32-bit architecture with an external 16-bit bus. The architecture also allows interfaces with 8-bit memories. However, with 8-bit memories, accesses to and from odd locations automatically access the higher-order byte. Thus, you would need external transceivers to access both even and odd locations. However, you can "trick" the processor and thereby save the space and cost associated with the external transceivers. Like everything else in life, the method doesn't come free—the price you pay is execution time.

The idea (**Figure 1**) is fairly simple: Connect the memory data bus to the least-significant bit (D0 to D7) of the µC. Then, connect the memory-address bus to the µC without using A0, so that the memory never sees an odd address. Thus, addressing the memory generates only even addresses. At first glance, the method might seem wasteful, because the memory occupies twice the space it needs, but with large memory spaces available (the V850 family can address as much as 16 Mbytes), the wastage should not present a problem. As an example, suppose register r2 contains the following data that you should store in a 1k×8 memory, starting at address 0x100:

Figure 1



A simple technique allows you to access odd memory locations without the need for external transceivers.

LISTING 1—SAMPLE PROGRAM TO ARRANGE STORAGE OPERATION

```
st.b    r2,0[r1]; 00 --> 0x100
shr     8,r2    ; shift 10 into the least significant byte
st.b    r2,2[r1]; 10 --> 0x101
shr     8,r2    ; shift 20 into the least significant byte
st.b    r2;4[r1]; 20 --> 0x102
shr     8,r2    ; shift 30 into the least significant byte
st.b    r2,6[r1]; 30 --> 0x103
```

Most significant byte			Least significant byte
30	20	10	00

After storing is complete, the memory resembles the following:

Address	Data
:	
:	
0x100	00
0x101	10
0x102	20
0x103	30
:	
:	

Assuming (r1)=0x100, the sample program in **Listing 1** arranges the storage operation. The tradeoff between extra hardware and longer execution time depends on the application's requirements. (DI #2366)

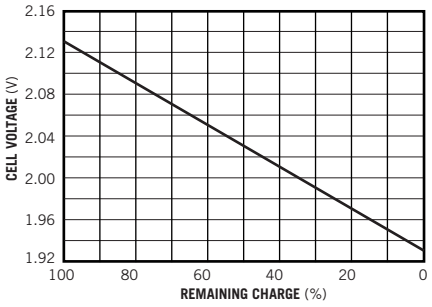
To Vote For This Design, Circle No. 328

Charge indicator gauges lead-acid batteries

Fran Hoffart, Linear Technology Corp, Milpitas, CA

Although rechargeable, sealed lead-acid cells are uncommon in portable applications, they are a good choice for standby applications, such as emergency lighting and burglar alarms. A key advantage to using these batteries is that you can determine the amount of remaining charge by measuring the open-circuit voltage. This technique is invalid for NiCd or NiMH cells. **Figure 1** shows the relationship between the amount of remaining charge versus the open-circuit battery voltage. This curve is accurate to approximately 10%, provided that you have not charged or discharged the battery for at least 24 hours. A simple circuit measures the open-circuit voltage, such as the expanded-scale voltmeter circuit in **Figure 2**, which follows the curve in **Figure 1**.

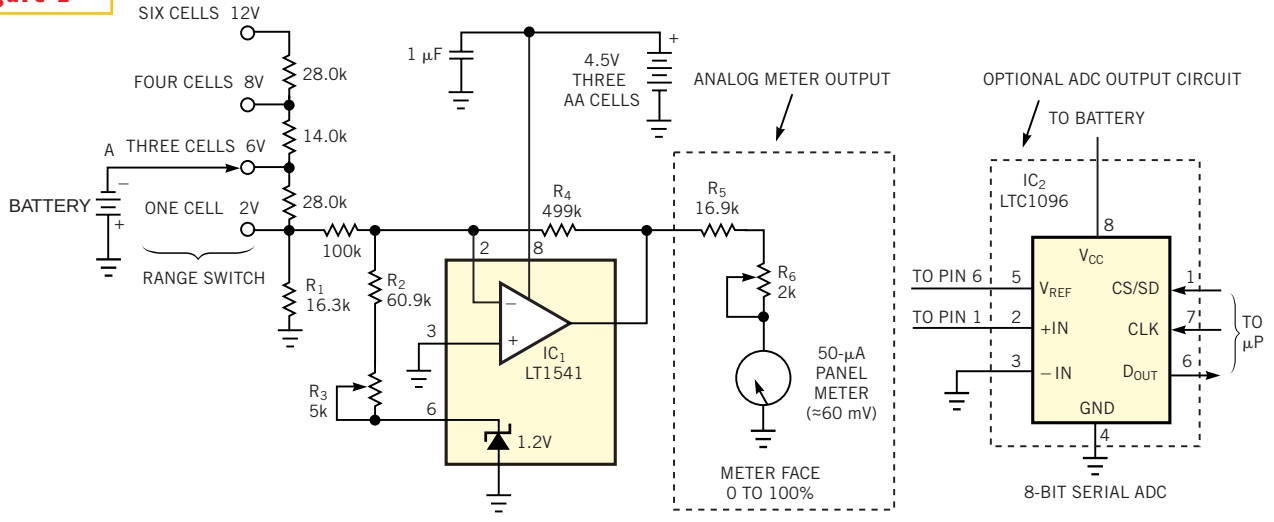
Figure 1



The curve of remaining charge versus open-circuit battery voltage for a sealed lead-acid battery is accurate to approximately 10%, if you haven't charged or discharged the battery for at least 24 hours.

Sealed lead-acid batteries are available in several sizes, from a single D size (2.5 Ahr) to multicell rectangular battery packs. These cells can provide high output currents and years of reliable backup power. Other desirable features include relatively simple charge requirements and low self-discharge. The low self-discharge and ease of determining the remaining charge make sealed lead-acid batteries an ideal choice for flashlights and portable lighting. The low self-discharge, which is approximately 5% per month at 25°C, means that a rechargeable flashlight using sealed lead-acid cells will still have usable

Figure 2



NOTES: USE 1% RESISTORS FOR STABILITY.
FOR R_1, SELECT A 16.2-kΩ RESISTOR THAT MEASURES HIGH.
THE VALUE OF R_5 VARIES WITH DIFFERENT METER MOVEMENTS.
WHEN USING THE OPTIONAL ADC OUTPUT CIRCUIT, CHANGE R_4 TO 600 kΩ.

To measure a sealed lead-acid battery's open-circuit voltage, an expanded-scale voltmeter circuit uses an op amp and reference to provide the necessary gain and offset to drive an analog or digital-panel meter, or optionally an ADC.

capacity of approximately 30% after one year of inactivity. NiCd and NiMH cells lose approximately 30% of their charge per month. A flashlight using NiCd cells requires a trickle charge when not in use to ensure reliable power when necessary. Without trickle charging, NiCd cells will completely discharge after three to four months of inactivity.

With the range switch in **Figure 1** in the one-cell position, the panel meter doesn't move until the input voltage exceeds 1.930V. Full scale corresponds to an input voltage of 2.130V. The op amp and reference provide the gain and offset for driving a digital panel meter, an ADC, or an analog meter with the meter scale calibrated from 0 to 100% of remaining charge. A rotary switch allows you to use the meter circuit with multicell battery packs containing one to six cells. You can measure other cell quantities by selecting the appropriate resistor divider values.

The circuit configures the op-amp section of IC_1, which also includes an unused comparator, as an inverting gain-of-five amplifier. This configuration produces a 1.000V change at the output for a 200-mV change at the input. The negative terminal of the battery connects to the op amp's inverting input resistor. To accomplish the 1.930V offset, IC_1's internal 1.200V reference, R_2, and R_3 generate a current that flows into the op amp's summing node (Pin 2). The op-amp output drives a standard 50-μA analog panel meter with a scale from 0 to 100%. You can also use a 1V full-scale digital panel meter or an ADC (**Figure 2**). The 8-bit ADC, IC_2, uses the 1.2V reference voltage of IC_1 for the ADC reference, giving a full-scale output (8 bits) for a 1.2V input. If you use the ADC, the op amp's gain must increase from 5 to 6 to provide an output of 1.2V from the op amp for a 200-mV change at the input. To make this change, you simply increase the value of R_4 to 600 kΩ. You can also use analog meters ranging from 100 μA to 1 mA, if you reduce the values of R_5 and R_6.

Calibrating the circuit requires an adjustable voltage source, preferably with coarse and fine voltage adjustment and a digital voltmeter. With three AA cells for power and the range switch in the one-cell position, apply a precise −2.130V to the input at point A. Connect a DVM to the op amp output (Pin 1) and adjust R_3 for a 1.000V reading on the DVM. Next, adjust R_6 for a full-scale reading, 100%, on the analog meter. Decreasing the voltage source by 100 mV to −2.030V should drop the DVM reading to 500 mV and drop the analog meter to midscale, or 50%. Dropping the voltage source an additional 100 mV to −1.930V results in a DVM reading near 0V and a corresponding meter indication of 0%. Because of minor resistor and offset-voltage errors, the output may not exactly equal 0V, but may be a few mV positive. For this application, this value is more than adequate. Resistor values of 1% provide the best accuracy and stability, but you can use a standard 16.2-kΩ 10% resistor that measures approximately 100Ω high for R_1. You can use **Table 1** to verify other ranges.

TABLE 1–BATTERY VOLTAGE VERSUS METER READING

Number of cells	Nominal voltage (V)	0%	50%	100%
1	2	1.93	2.03	2.13
3	6	5.79	6.09	6.39
4	8	7.72	8.12	8.52
6	12	11.58	12.18	12.78

The circuit does not require a power switch because the op-amp section of the circuit draws extremely low quiescent current (12 μA). Battery life should equal the shelf life of the battery, which is several years. The op amp's input also includes overvoltage and reverse-voltage protection. (DI #2359)

To Vote For This Design,
Circle No. 329

Use derivatives to catch RF calibration errors

Steven C Hageman, Hewlett-Packard Co, Santa Rosa, CA

Many RF-system calibrations involve checking for minimum power available or removing system offsets. One example is the checking of an RF source's output power. The system specifications may call for a minimum source power minus any cabling power loss, but a typical source may be able to provide more power than the manufacturer specifies as the minimum. To minimize the system cost, it is best to set the test-line limit to the minimum power plus a suitable instrumentation uncertainty (**Reference 1**).

This simplistic test may not catch all of the possible system problems. Loose RF connections or bad cables may result in power holes. Although these power holes may not always be deep enough to drop the power below the specified minimum test limit, no one wants to ship a system with a loose RF connection, or worse. Trained personnel may catch such a problem if they view it graphically, but this sort of test is very hard to quantify.

A better way to detect problems is to differentiate the data and place limits on the data's rate of change. This method is a surefire way to test for system problems that don't show up in the minimum-

power test. You can apply this technique to a large class of RF test and calibration issues, primarily the removal of system offsets during calibration. Many systems function properly with large offsets because calibration removes these offsets. However, the data usually has typical mismatch ripple effects over frequency, which cause the offset value to change as the frequency changes. If the offset ripple is too great or changes with frequency at a large rate, the stability of the calibration may be in jeopardy; a small change in the location of the ripple frequency may cause a large change in calibration offset data. By looking at the derivative data of the calibration, you can view the rate of change of the offset. When the rate of change reaches a certain level, the test alerts you, thereby the test is less subjective.

Assuming that your test is computer-controlled, you should be able to easily access the test data. You can apply the forward difference equation to the test data to find the first derivative on a point-by-point basis:

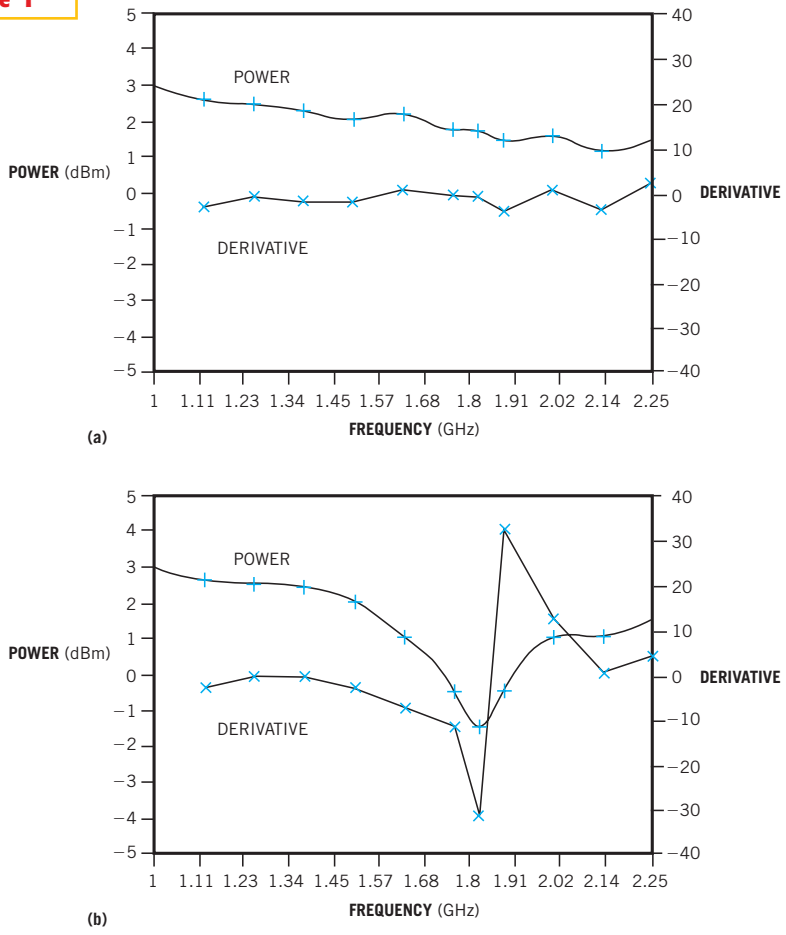
$$F\text{¢}x) = \frac{F(x + D x) - F(x)}{D x}$$

Figure 1a shows the results of a successful minimum-power test. The power is well above the −2-dBm limit and is well-behaved. The derivative data is also small. **Figure 1b** shows the same system when a connector is loose. In this case, the minimum power data is above the specification but is not well-behaved; in fact, a power hole appears. The power hole does not cause a failure in the minimum specified power, but remains a cause for concern for three reasons:

- a power hole results when a system problem in unstable with time, temperature, or shipping;
- as the loose connection moves, the null frequency may move in frequency, rendering subsequent calibrations useless;
- only a trained eye can determine from the plot that there is a failure.

You cannot determine the failure by simply looking at a pass/fail result. However, the derivative data from the power hole can generate a hard-fail indication.

Figure 1



In a well-behaved system (a), both the power data and its derivative are above the −2-dBm level. Tests of the same system with a loose connector (b) show that the power curve still doesn't dip below −2 dBm, but the derivative data indicates the existence of a power hole.

The derivative of the data in **Figure 1b** is 10 times the derivative data in **Figure 1a**. If you set the pass/fail criteria for the derivative data at 5 dB/Δx, the data from **Figure 1a** easily passes, and the data from **Figure 1b** fails.

You can use a small statistical base of measurements on different systems to set a qualified, statistical three-sigma limit on the derivative data. This limit will be insensitive to offset magnitude but will show point-to-point rate of change. Using the derivative data limits and the minimum power limits together will eliminate any chance of shipping an improperly functioning system to a customer. (DI #2360)

REFERENCE

1. Application Note AN64-1, "Fundamentals of RF and Microwave Power Measurements," Hewlett-Packard Co, Palo Alto, CA.

µC-based circuit performs frequency multiplication

Yongping Xia, Teldata Inc, Los Angeles, CA

The traditional frequency multiplier requires many elements: a phase comparator to detect the phase error between the input and the output signals, a lowpass filter to convert the phase error to a dc control signal, a VCO to generate the output, and a divider to set up the multiple ratio. The circuit in **Figure 1** uses a different approach to multiply frequency with a programmable multiple ratio from 1 to 7 (**Table 1**). Because the circuit is edge-triggered, the 50% output duty cycle is independent of the duty cycle of the input waveform. Test results show that the output frequency-range is

LISTING 1—FREQUENCY-MULTIPLIER CODE

```
.include "1200def.inc"
.device AT90S1200

.def        cnt         =r16            ;
.def        temp_1      =r17            ;
.def        temp_2      =r18            ;
.def        temp_3      =r19            ;
.def        temp_4      =r20            ;
.def        temp_5      =r21            ;
.def        temp_6      =r22            ;
.def        number      =r23            ;
.def        pulse       =r24            ;
.def        delay_1     =r25            ;
.def        delay_2     =r26            ;
.def        cnt_1       =r27            ;
.def        cnt_2       =r28            ;
                                        ;
reset:                                  ;
            rjmp  init                  ;
int:                                    ;
            in    temp_4,       PINB    ;     read input
            andi  temp_4,       $07     ;
            mov   temp_5,       temp_4;
            mov   temp_6,       temp_4;
            lsl   temp_5                ;
            lsr   temp_6                ;
            ldi   temp_2,       $0      ;
            cp    number,       temp_4;
            brne  next_1                ;
            ldi   temp_2,       $1      ;
next_1:                                 ;
            dec   temp_2                ;
            breq  next_2                ;
            out   PORTD,        pulse   ;
            inc   pulse                 ;
next_2:                                 ;
            cp    number,       temp_4;
            brlo  dec_cnt               ;
            cp    number,       temp_5;
            brlo  inc_cnt               ;
            sec                         ;
            rol   cnt_1                 ;
            rol   cnt_2                 ;
            brcc  int_out               ;
cnt_overflow:                           ;
            ldi   cnt_1,        $ff     ;
            ldi   cnt_2,        $ff     ;
            rjmp  int_out               ;
inc_cnt:                                ;
            add   cnt_2,        temp_3;
            breq  inc_1                 ;
            add   cnt_1,        cnt_2 ;
            brcc  inc_1                 ;
            inc   cnt_2                 ;
            breq  cnt_overflow          ;
   inc_1:                               ;
               inc   cnt_1              ;
               brne  int_out            ;
               inc   cnt_2              ;
               breq  cnt_overflow       ;
               rjmp  int_out            ;
   dec_cnt:                             ;
            cp    number,       temp_5;
            brlo  half_cnt              ;
            add   cnt_2,        temp_3;
            breq  dec_1                 ;
            sub   cnt_1,        cnt_2 ;
            brcc  dec_1                 ;
            subi  cnt_2,        $1      ;
            brcs  cnt_underflow         ;
   dec_1:                               ;
            subi  cnt_1,        $1      ;
            brcc  int_out               ;
            subi  cnt_2,        $1      ;
            brcc  int_out               ;
   cnt_underflow:                       ;
            ldi   cnt_1,        $1      ;
            ldi   cnt_2,        $0      ;
   int_out:                             ;
            ldi   number,       $0      ;
            rjmp  loop_1                ;
   half_cnt:                            ;
            clc                         ;
            ror   cnt_2                 ;
            ror   cnt_1                 ;
            rjmp  int_out               ;
   init:                                ;
            ser   temp_1                ;
            out   PORTB,        temp_1;
            out   PORTD,        temp_1;
            ldi   temp_1,       $0      ;
            out   DDRB,         temp_1;
            ldi   temp_1,       $1      ;
            out   DDRD,         temp_1;
            ldi   delay_1,      $ff     ;
            ldi   delay_2,      $ff     ;
            ldi   temp_1,       $40     ;
            out   GIMSK,        temp_1;
            ldi   temp_1,       $3      ;
            out   MCUCR,        temp_1;
            ldi   pulse,        $0      ;
            ldi   number,       $0      ;
            ldi   temp_3,       $0      ;
            in    temp_4,       PINB    ;
            andi  temp_4,       $07     ;
            mov   temp_5,       temp_4;
            mov   temp_6,       temp_4;
            lsl   temp_5                ;
            lsr   temp_6                ;
                                        ;
   loop_1:                              ;  endless loop
              sei                       ;
           mov   delay_1,      cnt_1    ;
           mov   delay_2,      cnt_2    ;
loop_2:                                 ;  delay loop
           subi  delay_1,      $1       ;
           brcc  loop_2                 ;
           subi  delay_2,      $1       ;
           brcc  loop_2                 ;
           cli                          ;
           out   PORTD,                 pulse ;  send output
           inc   pulse                  ;
           sbrs  pulse,                 $0    ;
           inc   number                 ;
           rjmp  loop_1                 ;
```

from 31 Hz to greater than 30 kHz.

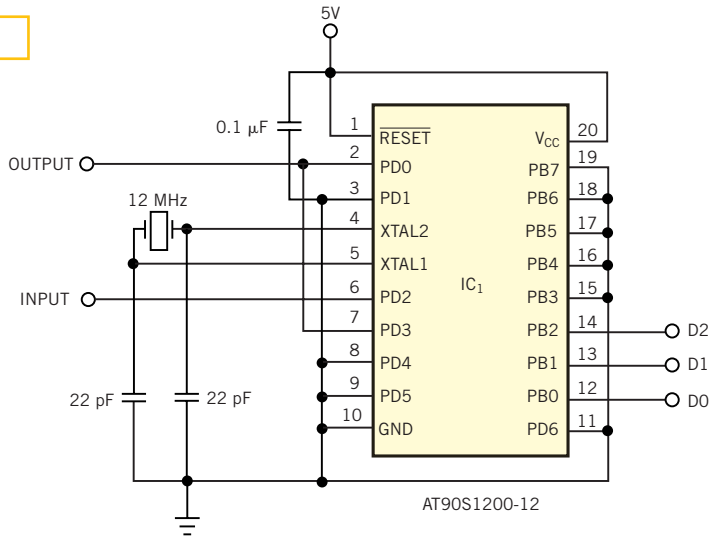
The AT90S1200 is a low-cost, high-speed μC, and most instructions need only one clock cycle. With a 12-MHz clock, these instructions take 83.3 nsec. This number places the high limit on the input frequency because software performs all functions. The program in **Listing 1** includes an endless loop to generate a square-wave output. The frequency of the output depends on the value of a 16-bit delay register that comprises two 8-bit registers: dly_1 and dly_2. The delay function is a countdown loop until it reaches zero. The larger the number in the delay register, the longer the delay time. The functions of the endless loop and the delay register are analogous to a VCO.

The AT90S1200 has an 8-bit counter whose input is the output signal. Because this counter is an up counter, the programmable multiple ratio loads into the counter in the 2's complement format. For instance, if the multiply ratio is four, the software loads that counter with 0xfc. Because the initial value of the counter is 0xfc, four output pulses cause the counter to overflow, which generates an interrupt. The function of this counter is analogous to the divider in a traditional frequency multiplier.

Every rising edge of the input signal also generates an interrupt. Thus, the interrupt subroutine must identify the events that trigger the interrupts. If the input causes the interrupt, the frequency of the output is too low. If the counter triggers the interrupt, the output frequency is too high. In both situations, the software must adjust the value of the delay register accordingly. The interruption subroutine is analogous to a phase comparator.

Listing 1 is available for downloading from *EDN*'s Web site, www.ednmag.com. Click on "Search Databases" and then enter the Software Center to download the file DI #2362.

Figure 1



A simple μC-based circuit can multiply frequency by 1 to 7.

TABLE 1–FREQUENCY-MULTIPLIER SETTINGS

D_2	D_1	D_0	Output frequency
0	0	0	–
0	0	1	x1
0	1	0	x2
0	1	1	x3
1	0	0	x4
1	0	1	x5
1	1	0	x6
1	1	1	x7

Kick start a crystal oscillator in Spice

Adam Chen, Cypress Semiconductor, Woodinville, WA

Starting up oscillator circuits and getting them to maintain oscillation in a Spice simulation is difficult. Some high-frequency crystal circuits require days for the oscillation to reach steady state. Thus, most designers separate the crystal's circuit simulation from the rest of the system design. How-

LISTING 1–SPICE NETLIST

```
Co     xtalin     xtalout     4pf
L1     xtalxi     1           9.076mH
C1     xtalout    3           13.613f
R1     1          2           25
Vsin   2          3           sin(0 0.7k 14.318Meg 0.1n 8e+6 0)
```

ever, a technique that gives a "kick" to an RLC equivalent circuit solves this problem. This method makes sure the simulation starts fast and quickly reaches the steady state.

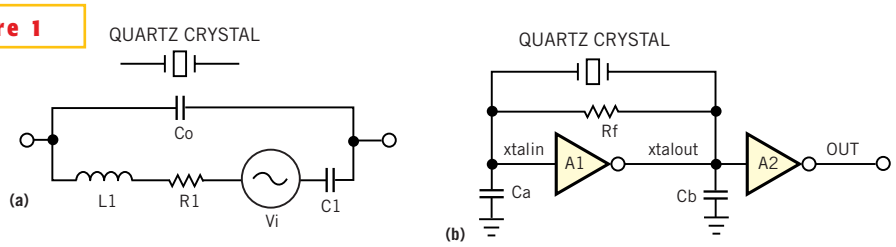
Figure 1a shows the equivalent RLC circuit of a quartz crystal. Most clock chips, such as Cypress Semiconductor's (www.cypress.com) CY227x and CY228x families, have a crystal circuit similar to **Figure 1b**. The circuit comprises the crystal, an inverter/gain block, and a feedback network.

Conventionally, Spice uses an initial condition for the RLC resonator, such as setting the inductor initial current to a certain value, to start the simulation. The reference frequency of the most common clock chips is 14.1318 MHz. The simulation takes a least a day to reach constant oscillation amplitude because high-Q resonators require long periods of time to reach a certain energy level.

The key to quickly starting this type of oscillator is giving a kick to the RLC equivalent circuit in the form of a high-voltage damped sinusoid that ultimately fades away. The frequency of this excitation is the expected frequency of the resonator. The source looks like a short circuit in the RLC circuit and does not alter any of the circuit's dc- bias conditions. **Figure 2** shows the simulation input/output waveforms, and **Figure 3** shows the excited sinusoidal voltage. The excited voltage is in the kV range because the voltages across L1 and C1 are in kV range when the LC tank is oscillating.

For a 14.318-MHz crystal, the equivalent circuit has Co=4 pF, C1=13.613 pF, L1=9.076 mH, and R1=25Ω. The excited voltage source is a simple Spice sinusoidal voltage source, Vsin in **Listing 1**. The Vsin statement includes the damping factor. (DI#2357)

Figure 1

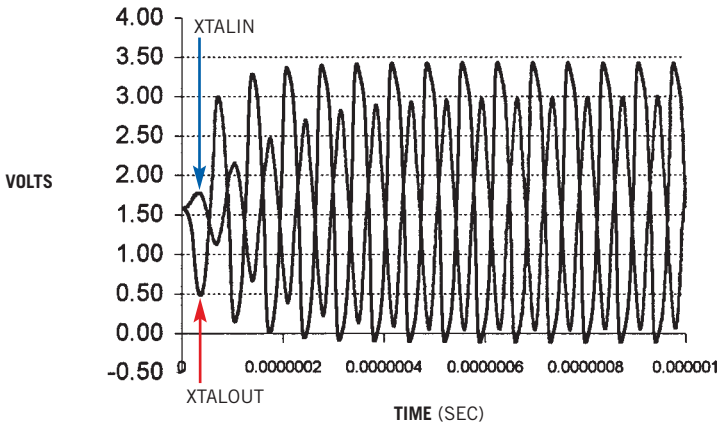


NOTES:
R1=EQUIVALENT RESISTANCE.
L1=EQUIVALENT INDUCTANCE.
C1=EQUIVALENT CAPACITANCE.
Co=INNER ELECTRODE CAPACITANCE.
Vi=EXCITED SINUSOIDAL VOLTAGE SOURCE.

NOTES:
Ca=Cb=34 pF, Rf=1MΩ.
QUARTZ CRYSTAL=14.318 MHz.
A1, A2 CMOS INVERTER.

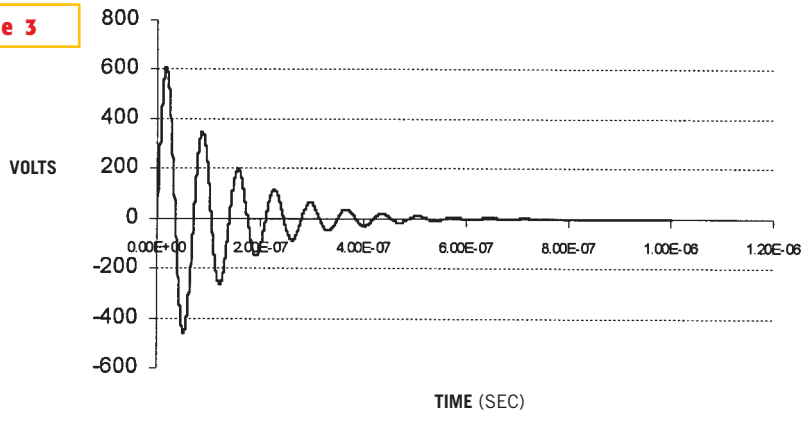
The equivalent circuit of a quartz crystal (a) includes equivalent resistance, R1; inductance, L1; capacitance, C1; and inner electrode capacitance, Co. In addition to the crystal, clock chips include an inverter/gain block and a feedback network (b).

Figure 2



The crystal circuit's simulation waveforms reach steady state in approximately 0.5 µsec.

Figure 3



An excited voltage source in the RLC circuit ensures oscillation startup and then quickly fades away.

To Vote For This Design, Circle No. 332